IEEE Xplore RELEASE 2.3

Sun, 10 Jun 2007, 12:22:00 AM EST

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Search Query Display

[redacted]

Recent Search Queries

#1	((averaging call trees xtree)<in>metadata)
#2	(call tree comparative profiles<IN>metadata)
#3	(call tree comparative profiles<in>metadata)
#4	(comparative profile<IN>metadata)
#5	(comparative profiles<IN>metadata)
#6	(trace data tree structure<IN>metadata)
#7	(trace<IN>metadata)
#8	(((trace<in>metadata))<AND>(performance<in>metadata))
#9	interprocedural analysis
#10	((interprocedural analysis)<AND>(profiling<in>metadata))
#11	((((interprocedural analysis)<and>(profiling<in>metadata)))<AND>(minimize<in>metadata))
#12	((interprocedural analysis)<AND>(profiling<in>metadata))
#13	((((interprocedural analysis)<and>(profiling<in>metadata)))<AND>(arcflow<in>metadata))
#14	(arcflow<IN>metadata)

[redacted]

Indexed by Inspec

Terms used
interprocedural analysis call graph performance metric aggregation

Found **122,905** of **1,035,585**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The Digital Library

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Modular interprocedural pointer analysis using access paths: design, implementation, and evaluation

Ben-Chung Cheng, Wen-Mei W. Hwu

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(855.37 KB) Additional Information: full citation, abstract, references, citings, index terms

In this paper we present a modular interprocedural pointer analysis algorithm based on access-paths for C programs. We argue that access paths can reduce the overhead of representing context-sensitive transfer functions and effectively distinguish non-recursive heap objects. And when the modular analysis paradigm is used together with other techniques to handle type casts and function pointers, we are able to handle significant programs like those in the SPECcint92 and SPECcint95 suites. W ...

2 Interprocedural parallelization analysis in SUIF

Mary W. Hall, Saman P. Amarasinghe, Brian R. Murphy, Shih-Wei Liao, Monica S. Lam

July 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 4

Publisher: ACM Press

Full text available: pdf(2.03 MB) Additional Information: full citation, abstract, references, citings, index terms

As shared-memory multiprocessor systems become widely available, there is an increasing need for tools to simplify the task of developing parallel programs. This paper describes one such tool, the automatic parallelization system in the Stanford SUIF compiler. This article represents a culmination of a several-year research effort aimed at making parallelizing compilers significantly more effective. We have developed a system that performs *full interprocedural parallelization analyses,* in ...

Keywords: Data dependence analysis, interprocedural data-flow analysis, parallelization, symbolic analysis

3 A schema for interprocedural modification side-effect analysis with pointer aliasing

Barbara G. Ryder, William A. Landi, Philip A. Stocks, Sean Zhang, Rita Altucher

March 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 2
Publisher: ACM Press

Full text available: pdf(1.72 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The first interprocedural modification side-effects analysis for C (MODC) that obtains better than worst-case precision on programs with general-purpose pointer usage is presented with empirical results. The analysis consists of an algorithm schema corresponding to a family of MODC algorithms with two independent phases: one for determining pointer-induced aliases and a subsequent one for propagating interprocedural ...

4 Interprocedural pointer alias analysis
Michael Hind, Michael Burke, Paul Carini, Jong-Deok Choi
July 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 4
Publisher: ACM Press

Full text available: pdf(502.42 KB) Additional Information: full citation, abstract, references, citings, index terms, review

We present practical approximation methods for computing and representing interprocedural aliases for a program written in a language that includes pointers, reference parameters, and recursion. We present the following contributions: (1) a framework for interprocedural pointer alias analysis that handles function pointers by constructing the program call graph while alias analysis is being performed; (2) a flow-sensitive interprocedural pointer alias analysis algorithm; (3 ...

Keywords: interprocedural analysis, pointer aliasing, program analysis

5 Visualizing the performance of higher-order programs
Oscar Waddell, J. Michael Ashley
July 1998 **ACM SIGPLAN Notices , Proceedings of the 1998 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering PASTE '98**, Volume 33 Issue 7
Publisher: ACM Press

Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

Profiling can provide the information needed to identify performance bottlenecks in a program, but the programmer must understand its relation to the program source in order to use this information. This is difficult due to the tremendous volume of data collected. Moreover, program transformations such as macro expansion and procedure inlining can obscure the relationship between the source and object code. Higher-order programs present additional challenges due to complex control flow and becau ...

6 Fast interprocedural class analysis
Greg DeFouw, David Grove, Craig Chambers
January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '98**
Publisher: ACM Press

Full text available: pdf(2.03 MB) Additional Information: full citation, references, citings, index terms

7 Exploiting hardware performance counters with flow and context sensitive profiling
Glenn Ammons, Thomas Ball, James R. Larus

May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation PLDI '97**, Volume 32 Issue 5
Publisher: ACM Press

Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, references, citings, index terms

A program profile attributes run-time costs to portions of a program's execution. Most profiling systems suffer from two major deficiencies: first, they only apportion simple metrics, such as execution frequency or elapsed time to static, syntactic units, such as procedures or statements; second, they aggressively reduce the volume of information collected and reported, although aggregation can hide striking differences in program behavior.This paper addresses both concerns by exploiting the har ...

8 Using types to analyze and optimize object-oriented programs
Amer Diwan, Kathryn S. McKinley, J. Eliot B. Moss
January 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 1
Publisher: ACM Press

Full text available: pdf(414.51 KB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented programming languages provide many software engineering benefits, but these often come at a performance cost. Object-oriented programs make extensive use of method invocations and pointer dereferences, both of which are potentially costly on modern machines. We show how to use types to produce effective, yet simple, techniques that reduce the costs of these features in Modula-3, a statically typed, object-oriented language. Our compiler performs type-based alias analysis to ...

Keywords: alias analysis, classes and objects, method invocation, object orientation, polymorphism, redundancy elimination

9 Vortex: an optimizing compiler for object-oriented languages
Jeffrey Dean, Greg DeFouw, David Grove, Vassily Litvinov, Craig Chambers
October 1996 **ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '96**, Volume 31 Issue 10
Publisher: ACM Press

Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, citings, index terms

Previously, techniques such as class hierarchy analysis and profile-guided receiver class prediction have been demonstrated to greatly improve the performance of applications written in pure object-oriented languages, but the degree to which these results are transferable to applications written in hybrid languages has been unclear. In part to answer this question, we have developed the Vortex compiler infrastructure, a language-independent optimizing compiler for object-oriented languages, with ...

10 On the usefulness of type and liveness accuracy for garbage collection and leak detection
Martin Hirzel, Amer Diwan, Johannes Henkel
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Publisher: ACM Press

Full text available: pdf(684.85 KB) Additional Information: full citation, abstract, references, citings, index terms

The effectiveness of garbage collectors and leak detectors in identifying dead objects

depends on the *accuracy* of their reachability traversal. Accuracy has two orthogonal dimensions: (i) whether the reachability traversal can distinguish between pointers and nonpointers (*type accuracy*), and (ii) whether the reachability traversal can identify memory locations that will be dereferenced in the future (*liveness accuracy*). This article presents an experimental study of the impo ...

Keywords: Conservative garbage collection, leak detection, liveness accuracy, program analysis, type accuracy

11 METRIC: Memory tracing via dynamic binary rewriting to identify cache inefficiencies

Jaydeep Marathe, Frank Mueller, Tushar Mohan, Sally A. Mckee, Bronis R. De Supinski, Andy Yoo

April 2007 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 29 Issue 2

Publisher: ACM Press

Full text available: pdf(912.62 KB) Additional Information: full citation, abstract, references, index terms

With the diverging improvements in CPU speeds and memory access latencies, detecting and removing memory access bottlenecks becomes increasingly important. In this work we present METRIC, a software framework for isolating and understanding such bottlenecks using partial access traces. METRIC extracts access traces from executing programs without special compiler or linker support. We make four primary contributions. First, we present a framework for extracting partial access traces based on ...

Keywords: Dynamic binary rewriting, cache analysis, data trace compression, data trace generation, program instrumentation

12 Article abstracts with full text online: A brief survey of program slicing

Baowen Xu, Ju Qian, Xiaofang Zhang, Zhongqiang Wu, Lin Chen

March 2005 **ACM SIGSOFT Software Engineering Notes**, Volume 30 Issue 2

Publisher: ACM Press

Full text available: pdf(535.10 KB) Additional Information: full citation, abstract, references, citings, index terms

Program slicing is a technique to extract program parts with respect to some special computation. Since Weiser first proposed the notion of slicing in 1979, hundreds of papers have been presented in this area. Tens of variants of slicing have been studied, as well as algorithms to compute them. Different notions of slicing have different properties and different applications. These notions vary from Weiser's syntax-preserving static slicing to amorphous slicing which is not syntax-preserving, an ...

Keywords: debugging, dependence analysis, pointer analysis, program analysis, program slicing

13 Program understanding: Analyzing feature implementation by visual exploration of architecturally-embedded call-graphs

Johannes Bohnet, Jürgen Döllner

May 2006 **Proceedings of the 2006 international workshop on Dynamic systems analysis WODA '06**

Publisher: ACM Press

Full text available: pdf(620.25 KB) Additional Information: full citation, abstract, references, index terms

Maintenance, reengineering, and refactoring of large and complex software systems are commonly based on modifications and enhancements related to features. Before

developers can modify feature functionality they have to locate the relevant code components and understand the components' interaction. In this paper, we present a prototype tool for analyzing feature implementation of large C/C++ software systems by visual exploration of dynamically extracted call relations between code components. T ...

Keywords: dynamic analysis, dynamic slicing, feature analysis, program comprehension, reverse engineering, software visualization

14 Dynamic metrics for java

Bruno Dufour, Karel Driesen, Laurie Hendren, Clark Verbrugge

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11

Publisher: ACM Press

Full text available: pdf(222.67 KB) Additional Information: full citation, abstract, references, citings, index terms

In order to perform meaningful experiments in optimizing compilation and run-time system design, researchers usually rely on a suite of benchmark programs of interest to the optimization technique under consideration. Programs are described as *numeric*, *memory-intensive*, *concurrent*, or *object-oriented*, based on a qualitative appraisal, in some cases with little justification. We believe it is beneficial to quantify the behaviour of programs with a concise and precisely ...

Keywords: Java, dynamic metrics, execution traces, optimization, profiling, program analysis, software metrics

15 An innovative low-power high-performance programmable signal processor for digital communications

J. H. Moreno, V. Zyuban, U. Shvadron, F. D. Neeser, J. H. Derby, M. S. Ware, K. Kailas, A. Zaks, A. Geva, S. Ben-David, S. W. Asaad, T. W. Fox, D. Littrell, M. Biberstein, D. Naishlos, H. Hunter

March 2003 **IBM Journal of Research and Development**, Volume 47 Issue 2-3

Publisher: IBM Corp.

Additional Information: full citation, abstract, references, cited by, index terms

We describe an innovative, low-power, high-performance, programmable signal processor (DSP) for digital communications. The architecture of this processor is characterized by its explicit design for low-power implementations, its innovative ability to jointly exploit instruction-level parallelism and data-level parallelism to achieve high performance, its suitability as a target for an optimizing high-level language compiler, and its explicit replacement of hardware resources by compile-time ...

16 Which pointer analysis should I use?

Michael Hind, Anthony Pioli

August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '00**, Volume 25 Issue 5

Publisher: ACM Press

Full text available: pdf(619.07 KB) Additional Information: full citation, abstract, references, citings, index terms, review

During the past two decades many different pointer analysis algorithms have been published. Although some descriptions include measurements of the effectiveness of the algorithm, qualitative comparisons among algorithms are difficult because of varying

infrastructure, benchmarks, and performance metrics. Without such comparisons it is not only difficult for an implementor to determine which pointer analysis is appropriate for their application, but also for a researcher to know which algo ...

Keywords: data flow analysis, interprocedural pointer analysis

17 Software unit test coverage and adequacy

Hong Zhu, Patrick A. V. Hall, John H. R. May
December 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 4
Publisher: ACM Press

Full text available: pdf(477.42 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Objective measurement of test quality is one of the key issues in software testing. It has been a major research focus for the last two decades. Many test criteria have been proposed and studied for this purpose. Various kinds of rationales have been presented in support of one criterion or another. We survey the research work in this area. The notion of adequacy criteria is examined together with its role in software dynamic testing. A review of criteria classification is followed by a sum ...

Keywords: comparing testing effectiveness, fault detection, software unit test, test adequacy criteria, test coverage, testing methods

18 Online Cycle Detection and Difference Propagation: Applications to Pointer Analysis

David J. Pearce, Paul H. J. Kelly, Chris Hankin
December 2004 **Software Quality Control**, Volume 12 Issue 4
Publisher: Kluwer Academic Publishers

Full text available: Publisher Site Additional Information: full citation, abstract

This paper presents and evaluates a number of techniques to improve the execution time of interprocedural pointer analysis in the context of C programs. The analysis is formulated as a graph of set constraints and solved using a worklist algorithm. Indirections lead to new constraints being added during this procedure. The solution process can be simplified by identifying cycles, and we present a novel online algorithm for doing this. We also present a difference propagation scheme which avoi ...

19 Continuous program optimization: A case study

Thomas Kistler, Michael Franz
July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4
Publisher: ACM Press

Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

20 Accurate static estimators for program optimization

Tim A. Wagner, Vance Maverick, Susan L. Graham, Michael A. Harrison

June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation PLDI '94**, Volume 29 Issue 6

Publisher: ACM Press

Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references, citings, index terms

Determining the relative execution frequency of program regions is essential for many important optimization techniques, including register allocation, function inlining, and instruction scheduling. Estimates derived from profiling with sample inputs are generally regarded as the most accurate source of this information; static (compile-time) estimates are considered to be distinctly inferior. If static estimates were shown to be competitive, however, their convenience would outweigh minor ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next .

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	12	performance adj analysis with (calltree call adj tree)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 15:52
S2	2214	graph with cost	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 15:52
S3	10	graph with cost and (calltree call adj tree call adj graph call adj hierarchy)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:06
S4	2	"6338159".pn.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:08
S5	575	dynamic near3 (trac$3 profil$3) same reduc$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:09
S6	614	dynamic near3 (trac$3 profil$3) same reduc$5	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:09
S7	48	dynamic near3 (trac$3 profil$3) same reduc$5 and (performance with (analysis analyz$4))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:10

EAST Search History

S8	3	dynamic near3 (trac$3 profil$3) same reduc$5 and (performance with (analysis analyz$4)) and (callgraph call adj graph calltree call adj tree)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:50
S9	22	dynamic near3 (trac$3 profil$3) and (performance with (analysis analyz$4)) and (callgraph call adj graph calltree call adj tree)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:51
S10	14	dynamic near3 (trac$3 profil$3) and (performance with (analysis analyz$4)) and (callgraph call adj graph calltree call adj tree) and pass	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 16:51

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L2	151	arc adj tool arcflow adj tool callgraph adj tool	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:26
L3	151	arc adj tool arcflow adj tool	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:26
L4	7193	717/124-133,144,151-171.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:29
L5	171	717/132,133.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:29
L6	701	717/154-157.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:30
L7	108	717/132.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:30
L8	81	717/133.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:30

EAST Search History

L9	1	717/133.ccls. and 2	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:30
L10	18	(calltree call adj tree call adj graph call adj hierarchy) and 8	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:31
L11	1144	(calltree call adj tree call adj graph call adj hierarchy program adj flow) and ((first and second) multiple two) with (executions passes runs)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 17:46
L12	23	(calltree call adj tree call adj graph call adj hierarchy program adj flow) and ((first and second) multiple two) with (executions passes runs) with (averag$3 total$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:01
L13	51	(calltree call adj tree call adj graph call adj hierarchy program adj flow) with minimiz$4	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:01
L14	41	(calltree call adj tree call adj graph call adj hierarchy program adj flow) with minimiz$4 and performance	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:01
L15	12	(calltree call adj tree call adj graph call adj hierarchy program adj flow) with minimiz$4 and performance and (profil$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:05

EAST Search History

L16	4273	(application program system) near3 performance and (trac$4 profil$4) with (dynamic$5)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:07
L17	331	(application program system) near3 performance and (trac$4 profil$4) with (dynamic$5) and (call adj (graph tree) program adj (flow) flow adj graph)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:08
L18	273	(application program system) near3 performance and (trac$4 profil$4) with (dynamic$5) and (call adj (graph tree) program adj (flow) flow adj graph) and (minimiz$4 averag$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:08
L19	106	(application program system) near3 performance and (trac$4 profil$4) with (dynamic$5) and (call adj (graph tree) program adj (flow) flow adj graph) and (minimiz$4 averag$4) with (trac$4 profil$4 runs)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:09
L20	60	(application program system) near3 performance and (trac$4 profil$4) with (dynamic$5) and (call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) with (trac$4 profil$4 runs)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:09
L21	2	(application program system) near3 performance and (trac$4 profil$4) with (dynamic$5) and (call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) with (trac$4 profil$4 runs) same minimiz$4	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:20
L22	5	(application program system) near3 performance and (call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) with (trac$4 profil$4 runs) same minimiz$4	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:22

EAST Search History

L23	3	22 not 21	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:21
L24	131	(call adj (graph tree) program adj (flow) flow adj graph) with (averag$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:22
L25	5	(call adj (graph tree) program adj (flow) flow adj graph) with (averag$4) adj out	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:23
L26	1	(call adj (graph tree) program adj (flow) flow adj graph) with (averag$4) same variation	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:23
L27	172	(call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) same variation	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:23
L28	2	(call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) adj out same variation	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:24
L29	62	(call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) adj out	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:24

EAST Search History

L30	39	(call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) adj out and (profil$4 trac$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:28
L31	25	(call adj (graph tree) program adj (flow) flow adj graph) and (averag$4) near3 node and (profil$4 trac$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:37
L32	270	(call adj (graph tree) program adj (flow) flow adj graph calltree) with (averag$4 merg$3 combin$3 minimiz$3) and (profil$4 trac$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:38
L33	44	(call adj (graph tree) program adj (flow) flow adj graph calltree) with (averag$4 merg$3 combin$3 minimiz$3) and (profil$4 trac$4) and add$3 with node	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:53
L34	4	32 and 3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:53
L35	554	profiling adj tool	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:53
L36	122	profiling adj tool and (trace)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:53

EAST Search History

L37	35	profiling adj tool and (trace) and averag$3	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/10 18:54

Web Results **1 - 10** of about **9,850,000** for **variations in program flow**. (**0.12** seconds)

Evaluating **variations** on **program** slicing for debugging (data-**flow** ...
Daniel Kästner , Stephan Wilhelm, Generic control **flow** reconstruction from assembly code, ACM SIGPLAN Notices, v.37 n.7, July 2002 ...
portal.acm.org/citation.cfm?id=911996&dl=ACM&coll=portal - Similar pages

book **variations** on constants: **flow** analysis of sequential parallel ...
Variations on constants: **flow** analysis of sequential & parallel **programs** (Lecture notes in computer science, Vol. 3800). Author(s) : MüLLER-OLM Markus ...
www.lavoisier.fr/notice/gb332765.html - 19k - Cached - Similar pages

Variations on Constants : **Flow** Analysis of Sequential and Parallel ...
Variations on Constants : **Flow** Analysis of Sequential and Parallel **Programs** only $62.29, get the **Variations** on Constants : **Flow** Analysis of Sequential and ...
www.bestprices.com/cgi-bin/vlink/3540453857?id=nsession - 22k - Cached - Similar pages

[PDF] SIEVE: A TOOL FOR AUTOMATICALLY DETECTING **VARIATIONS** ACROSS ...
File Format: PDF/Adobe Acrobat - View as HTML
about a **program** execution's control and data-**flow** behav- ... late **variations in program** versions, we abstract a **program**. as a sequence of memory reads and ...
www.cs.purdue.edu/research/technical_reports/2005/TR%2005-019.pdf - Similar pages

Heat-**flow variations** correlated with buried basement topography on ...
SYSTEMATIC heat-**flow variations** are commonly observed on sedimented flanks of mid-ocean ridges ... Ocean Drilling **Program**, Sci. Results 109, (in the press). ...
www.nature.com/nature/journal/v342/n6249/abs/342533a0.html - Similar pages

Digital Systems Research Center: Note 1997-032
A Semantic Approach to Secure Information **Flow** ... applied to reasoning about indirect leaking of information through **variations in program** behavior (e.g., ...
gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1997-032.html - 4k - Cached - Similar pages

Computation of the compensating **variation** within a random utility ...
The **program flow** and the **program** list with comments are supplied. ... in particular in term of Compensating **Variation** (CV), within a random utility model. ...
ideas.repec.org/p/wpc/wplist/wp02_06.html - 9k - Cached - Similar pages

[PDF] Sieve: A Tool for Automatically Detecting **Variations** Across ...
File Format: PDF/Adobe Acrobat
Sieve: A Tool for Automatically Detecting **Variations** Across **Program** Versions .... **flow** behavior, new techniques are still required to precisely ...
doi.ieeecomputersociety.org/10.1109/ASE.2006.61 - Similar pages

Metering Research Facility **Program** : Effects of diameter mismatch ...
Metering Research Facility **Program** : Effects of diameter mismatch and line pressure **variations** on ultrasonic gas **flow** meter performance Grimley, Terrence A. ...
bibliotheek.eldoc.ub.rug.nl/root/RuGbreed/metering/?FullItemRecord=ON - 17k - Cached - Similar pages

Experimental study of the perturbation of the **flow** of current to a ...
Variation of the **flow** of primary current to a disk electrode due to an insulating or ... A **program**, developed on Testpoint software, controls the impedance ...
www.mtm.kuleuven.be/Research/SURF/webcl.htm - 6k - Cached - Similar pages

1 2 3 4 5 6 7 8 9 10 **Next**

---

variations in program flow Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

---